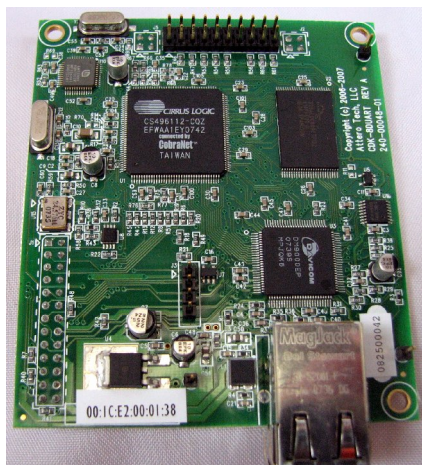


CDK-8DUART

OEM CobraNet Module



Design Guide

Date 1/17/2012

Revision 03

Contents

1 – Overview	1
1.1 – Notes on Modules	1
2 – Digital Audio Interface Connectivity.....	2
2.1 – Pin Descriptions	2
2.1.1 - Audio Clocks	2
2.1.2 – Digital Audio	2
2.1.3 – Serial Bridge	3
2.1.4 – Control	3
2.1.5 – Power	3
2.2 – Reference Design	3
3 – PCB Design Guidelines.....	12
3.1 – Power Supply.....	12
3.2 – Clock Termination.....	12
3.3 – Clock Signal Parallel End Termination	12
3.4 – Grounding	12
3.5 – Signal Separation	12
3.6 – Chassis Grounding.....	12
4 – End Product Bring-up Hints	13
4.1 – Power Supplies	13
4.2 – Clocks and I2S Lines	13
4.3 – Ethernet LED Activity.....	13
APPENDIX A – CDK-8DUART SHMI Message Protocol.....	A-1
A1 – Terminology.....	A-1
A2 – System Description	A-2
A3 – Serial Format.....	A-3
A3.1 – Message Format.....	A-3
A3.2 – Read Messages	A-3
A3.3 – Write Messages	A-5
A3.4 – Baud Rate Change Message.....	A-6
A3.5 – <i>Nack</i> Response Descriptions	A-7
A4 – Commands & Responses	A-8
A4.1 – Legend	A-9
A4.2 – sysDescr.....	A-9
A4.3 – sysContact	A-10
A4.4 – sysName.....	A-10
A4.5 – sysLocation	A-10
A4.6 – ifPhysAddress	A-10
A4.7 – flashPersistEnable.....	A-11
A4.8 – errorCode.....	A-11
A4.9 – errorCount	A-11
A4.10 – modeRateControl.....	A-11
A4.11 – conductorStatus.....	A-12
A4.12 – serialFormat	A-12
A4.13 – serialBaud	A-12
A4.14 – serialRxMac.....	A-13
A4.15 – serialTxMac.....	A-13
A4.16 – rxSubFormat	A-13
A4.17 – rxBundle	A-14
A4.18 – rxSubMap.....	A-14
A4.19 – txBundle.....	A-15
A4.20 – txSubCount.....	A-15
A4.21 – txUnicastMode.....	A-15
A4.22 – txMaxUnicast.....	A-16
A4.23 – txSubMap	A-16
A4.24 – txSubFormat	A-16
A4.25 – ipMonCurrentIP.....	A-17
A4.26 – procMode.....	A-18
A4.27 – stdUserVersionMajor	A-18

A4.28 - stdUserVersionMinor	A-18
A4.29 - stdUserString	A-18
A4.30 - stdUserInteger	A-19
APPENDIX B - Introduction to CobraNet	B-1
APPENDIX C - Reference Documents	C-1

1 – Overview

This Design Guide has been assembled to aid in the integration of the CDK-8DUART CobraNet-enabled module for OEM applications.

This guide contains schematic examples for connecting the audio interfaces, schematics, and sample code showing various methods of connecting the HMI control interface, PCB layout guidelines, and also some end-product bring up hints and tips. For those not familiar with CobraNet, there is also a brief overview of CobraNet and its features in 4.3APPENDIX B.

1.1 – Notes on Modules

The CDK-8DUART is shipped with the DSP processing bypassed and they contain no DSP customization whatsoever. Essentially, they are a blank canvas. This may be suitable for your application as is but it is possible you will want to use the on-board user DSP to manipulate the signals within the device. If that is the case, additional design steps are required to customize the CDK-8DUART.

If you do intend to customize the module, you will need to obtain DSP Conductor software from the Cirrus website. This software allows the user to create custom designs that can be downloaded to the CobraNet chip. However, it should be noted that DSP Conductor only loads the design into RAM within the CobraNet part. When the device is turned off, the design will be lost. The CDK-8 is capable of retaining customized designs but in order to do so, those customizations have to be part of the firmware. Please contact Sales@atterotech.com to discuss possible options.

All CobraNet ICs, including the ones used on the CDK-8DUART, can retain all of the common CobraNet variables such as bundle numbers during power off by setting the Persistence parameter. However, the Persistence parameter does not affect the settings of any of the parameters used by the user DSP in a custom design as there is no way to predict what variables will exist for any given user design. There is a separate method for storing those parameters in the flash so they are not lost. The tools necessary to do this can be obtained via your local Cirrus distributor or your local Cirrus sales representative.

2 - Digital Audio Interface Connectivity

The audio connector on the CDK-8DUART contains all the clocks and digital signals needed for up to eight channels of digital audio as well as connections for the serial bridge interface and various control signals such as a reset input and a watchdog output. The CDK-8DUART module also takes power through this connector. Below is a schematic diagram of the audio connector.

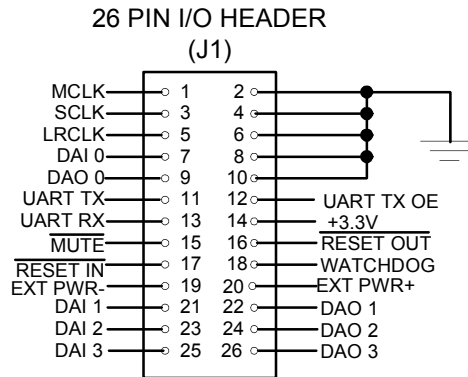


Figure 1 - Digital Audio Connector Pin-out

2.1 - Pin Descriptions

2.1.1 - Audio Clocks

Signal	Type	Description
CDK_MCLK	Output	Digital audio master clock generated by the CDK-8
CDK_SCLK	Output	Digital audio sample clock generated by the CDK-8
CDK_LRCLK	Output	Digital audio left/right clock generated by the CDK-8.

2.1.2 - Digital Audio

Signal	Type	Description
DAI1_DATA0	Input	Digital audio input bit stream for channels 1 and 2*
DAI1_DATA1	Input	Digital audio input bit stream for channels 3 and 4*
DAI1_DATA2	Input	Digital audio input bit stream for channels 5 and 6*
DAI1_DATA3	Input	Digital audio input bit stream for channels 7 and 8*
DAO1_DATA0	Output	Digital audio output bit stream for channels 1 and 2*
DAO1_DATA1	Output	Digital audio output bit stream for channels 3 and 4*
DAO1_DATA2	Output	Digital audio output bit stream for channels 5 and 6*
DAO1_DATA3	Output	Digital audio output bit stream for channels 7 and 8*

* Digital audio is setup as I2S by default. CobraNet devices also support two other SSI formats (see the relevant device manual for details). If one of these is required instead of I2S, an alteration to the default firmware will be needed. Contact Sales@atterotech.com for more details.

2.1.3 – Serial Bridge

Signal	Type	Description
BRIDGE_UART_RXD	Input	Serial bridge receive line
BRIDGE_UART_TXD	Output	Serial bridge transmit line
BRIDGE_UART_TX_OE	Output	Enable transmit (active high) drive for two wire multi-drop interface.

2.1.4 – Control

Signal	Type	Description
$\overline{\text{MUTE}}$	Output	Asserts (active low) during initialization and when a fault is detected or connection to the network is lost.
$\overline{\text{RESET_IN}}$	Input	Active low external reset control for the CDK-8 module.
$\overline{\text{RESET_OUT}}$	Output	Active low reset output. A combination of the RESET_IN signal and an onboard brown out reset detector.
WATCHDOG	Output	Toggles at 750 Hz nominal rate to indicate proper operation. Period duration in excess of 200ms indicates hardware or software failure has occurred and the interface should be reset. Note that improper operation can also be indicated by short pulses (<100 ns).

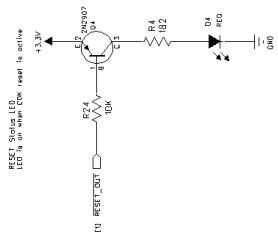
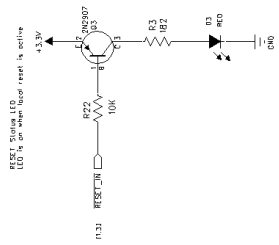
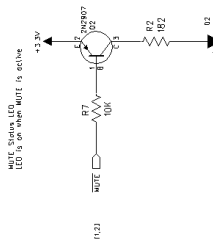
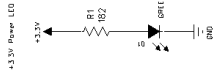
2.1.5 – Power

Signal	Type	Description
+3.3V	Power	+3.3 V \pm 0.3 V, 500 mA Typ., 750 mA Max
GND	Power	
EXT_PWR+	Output	These connections provide the potential to obtain power via the unused pairs of the CAT-5 network cable.
EXT_PWR-	Output	

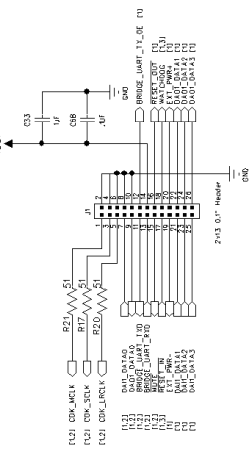
2.2 – Reference Design

The reference design includes a DAC and an ADC connected to the audio interface of the CDK-8DUART. The CDK-8DUART is capable of eight channels, though only two are used in the reference design. Electronic copies of the designs can be made available on request.

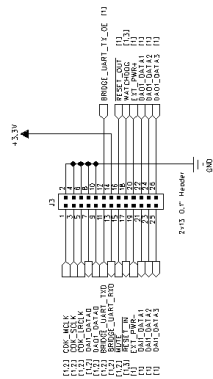
Status LEDs



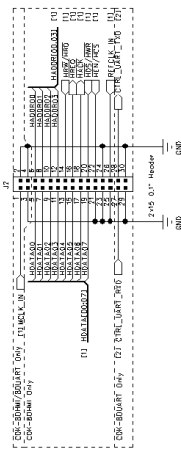
CDK-B Module Digital I/O Header



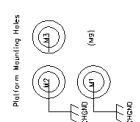
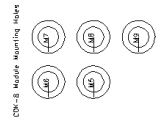
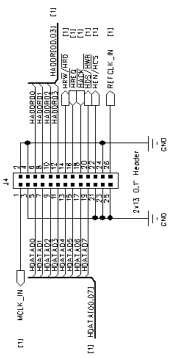
User Digital I/O Header



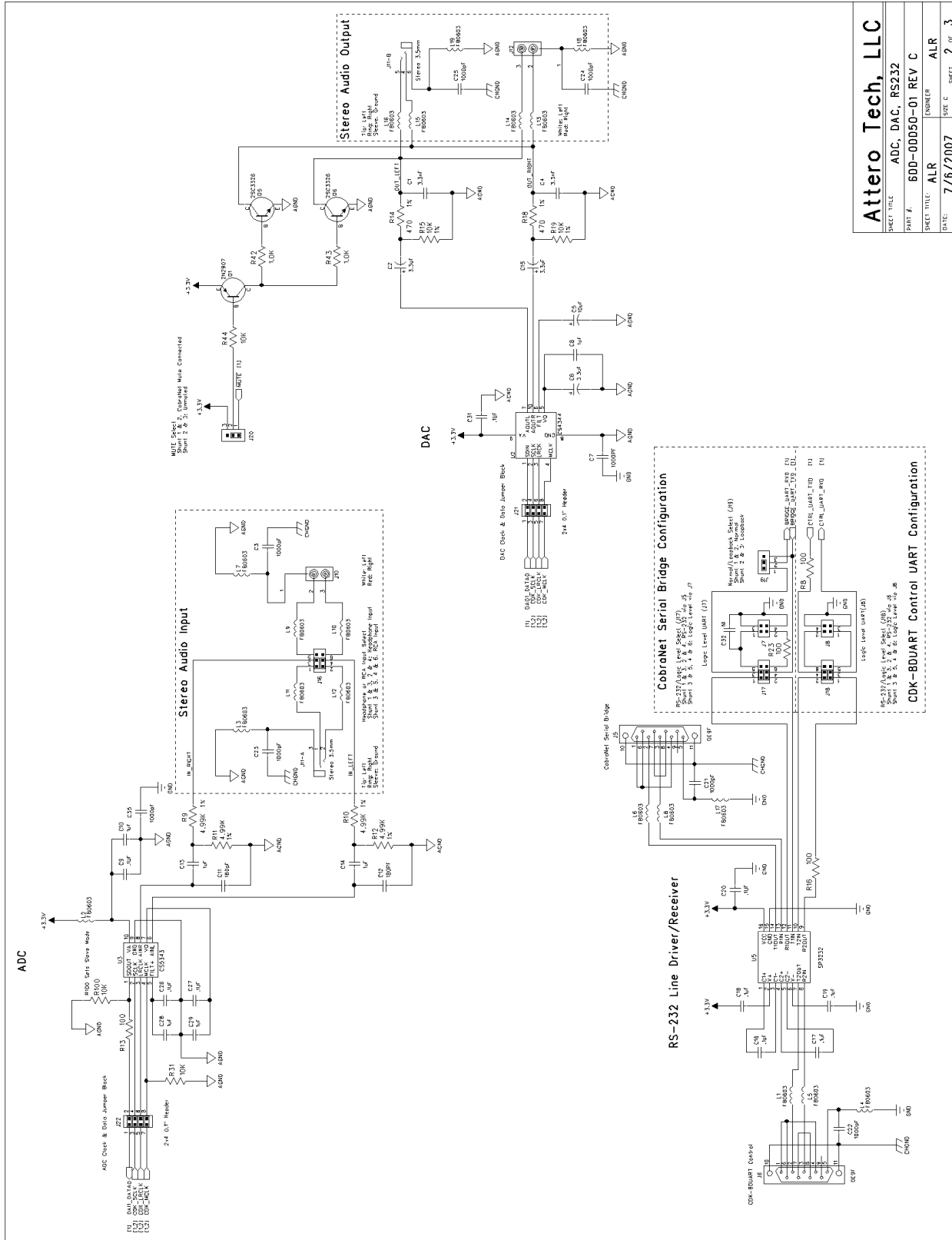
CDK-8DHWI/8DUART Module HWI/UART Header



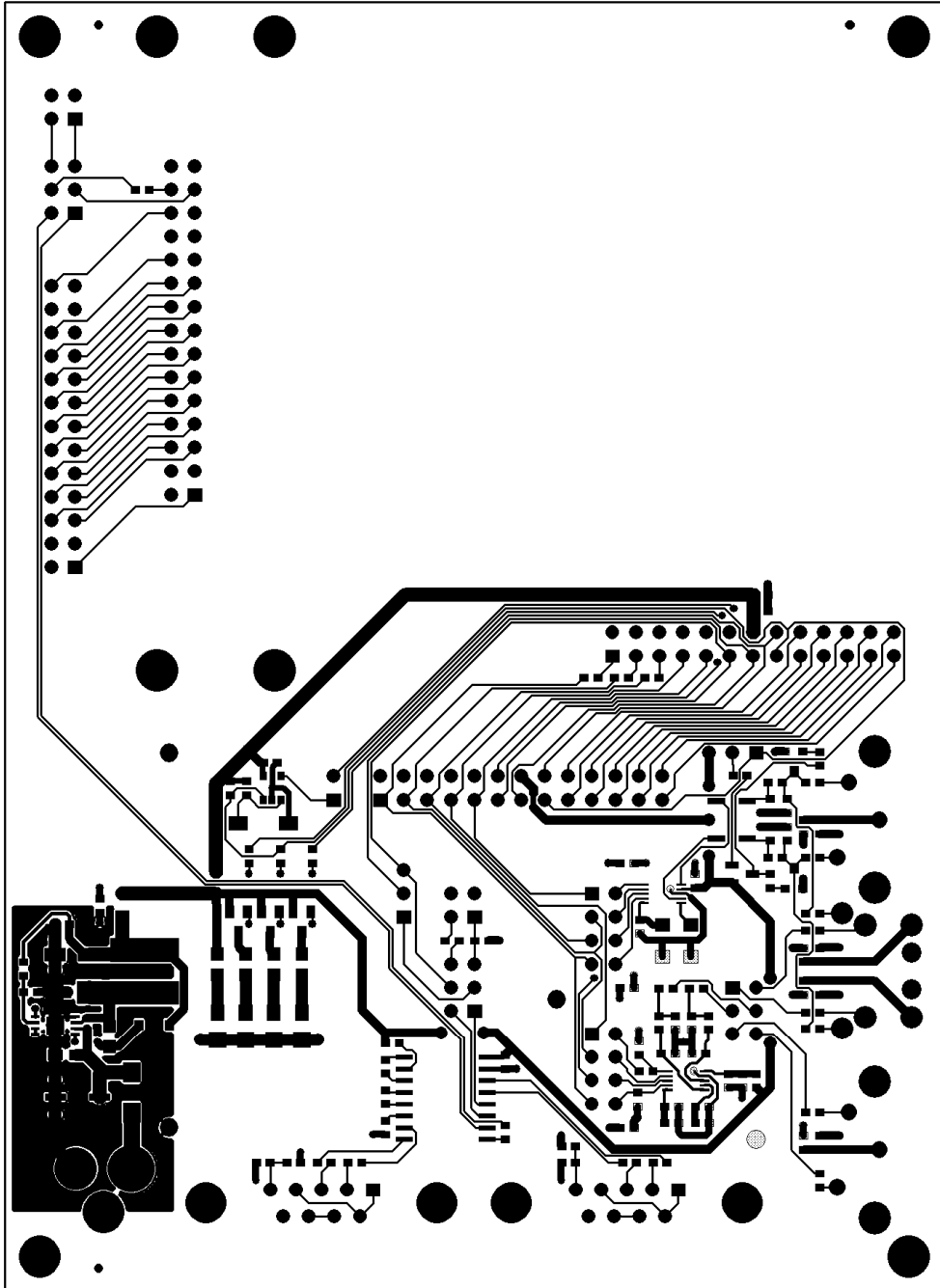
User HWI Header



Attero Tech, LLC	
Headers, LEDs	
PART #	600-00050-01 REV C
SHEET TITLE	ALR ENHANCER
DATE	7/6/2007
SHEET	1 OF 3

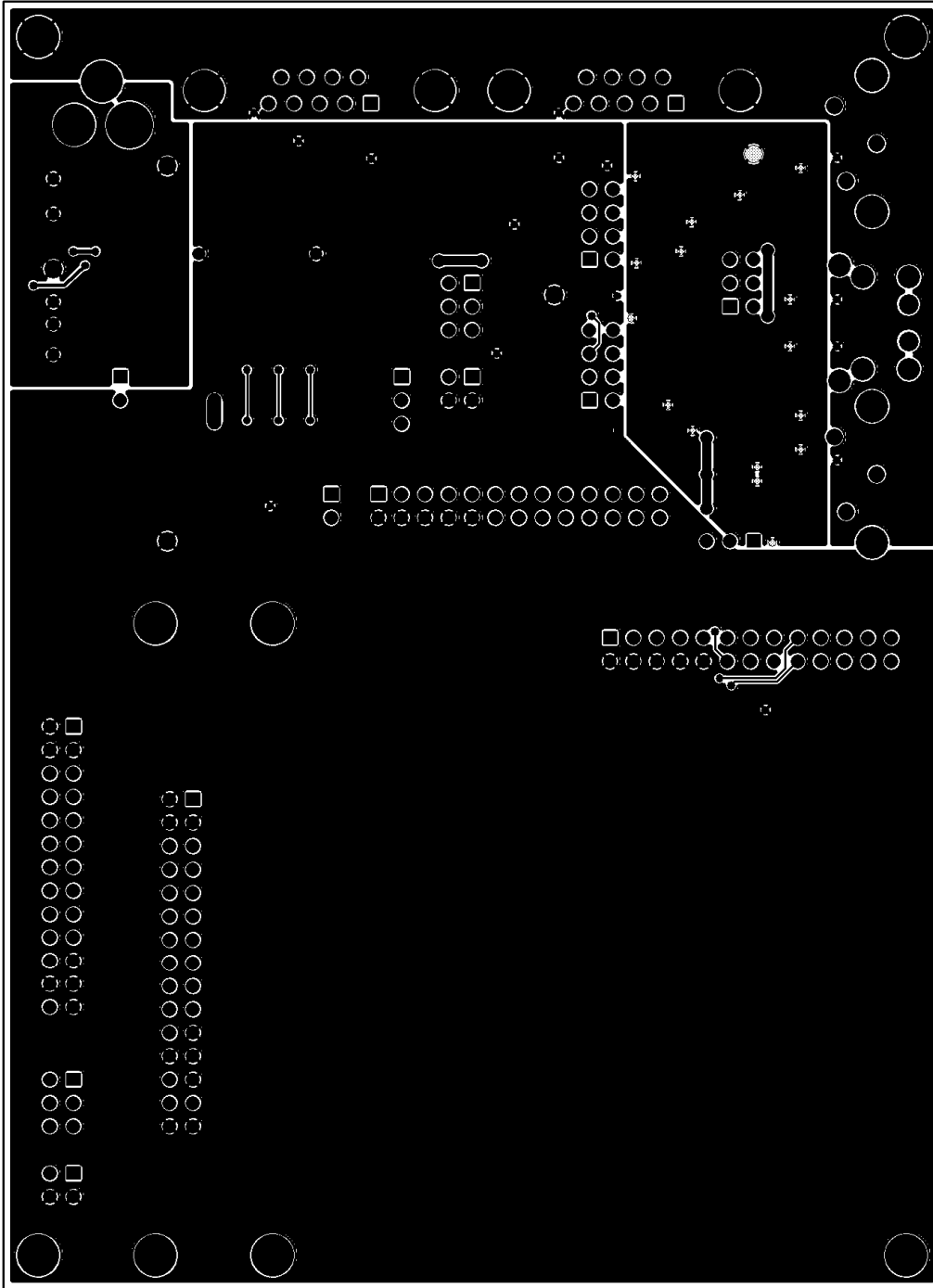


Attero Tech, LLC	
SHEET TITLE	ADC, DAC, RS232
PART #	600-00050-01 REV C
SHEET TITLE	ALR
DATE	7/6/2007
DESIGNER	ALR
SIZE	C
SHEET	2 OF 3



Top Copper

Figure 3 - PCB Top Copper Layer



Bottom Copper

Figure 4 - PCB Bottom Copper Layer (viewed from below)

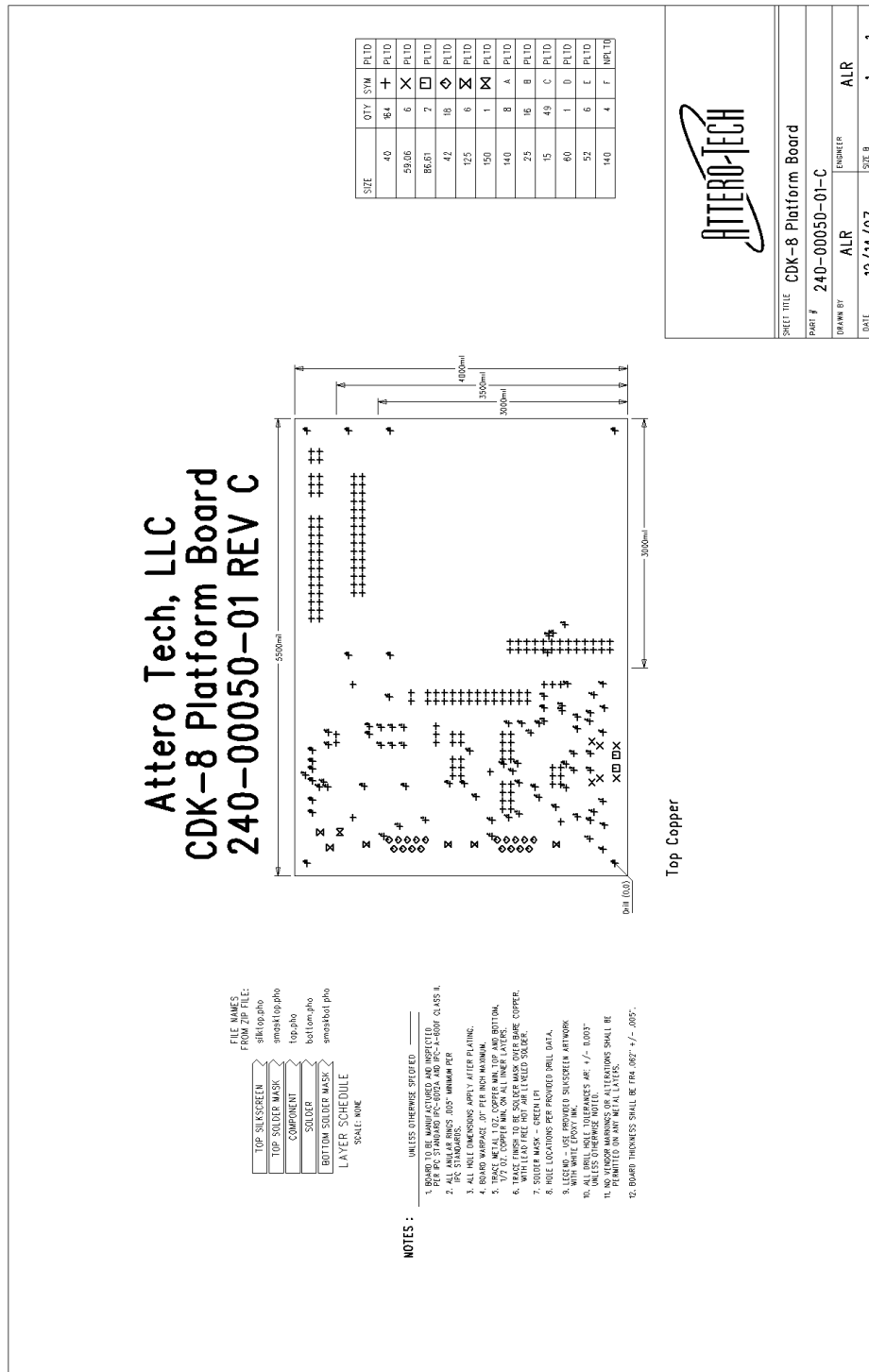


Figure 5 - PCB Assembly Notes

This page is intentionally blank

3 – PCB Design Guidelines

In order to ensure the best signal integrity and minimize EMI radiation problems, adhere to the following guidelines when designing your PCB.

3.1 – Power Supply

It is recommended that bypass capacitors be placed near +3.3V power pin (pin 12) to reduce the AC signal return path. A 0.1uF capacitor in parallel with a 10uF - 47uF electrolytic between +3.3V and ground are suggested.

3.2 – Clock Termination

All audio clock sources should be terminated with series source termination resistors to minimize ringing. In general, a value of 75 to 100 ohms has been determined to provide adequate source termination without causing any adverse effects. The audio clocks should be routed as direct and short as possible, ideally without any vias. If vias are required, keep them to a minimum.

3.3 – Clock Signal Parallel End Termination

In designs where the clock signals are long by necessity, thought should be given to terminating the lines. In general this should be with a small capacitor (in the order of 1000pF or smaller) in series with a small resistor (around 75 ohms) between the clock and ground. The precise values used will depend on the trace impedance of the terminated trace.

3.4 – Grounding

It is imperative that the ground plane referenced by any of the clocks interfacing to the CDK module be located below the signal trace. In other words, avoid any discontinuities or cutouts in the ground plane when routing clock signals over the plane. This helps to minimize the total loop area of the signal and reduces the overall affects of EMI.

3.5 – Signal Separation

Any clocks should be adequately separated. Parallel routing of the signals for significant distances should also be avoided. In some cases, space requirements may force the clocks to be routed in near proximity, in this case it is also recommended that the signal ground be poured between the clock traces on the signal layer to provide isolation between the clocks and minimize signal coupling from trace to trace.

3.6 – Chassis Grounding

Due to the various clock signals required to implement a CobraNet interface, it is imperative that chassis grounding be given consideration for integration into end products. The CDK-8DUART module has a chassis ground connection through the lower right mounting hole. A solid galvanic connection to the main chassis ground/enclosure is effective in helping to reduce emissions. This should be especially noted for any device requiring regulated emissions certification.

4 – End Product Bring-up Hints

Basic functionality can be tested before integrating into the target product by applying a regulated and current limited +3.3V power supply between J1 pin 14 (+3.3V) and J1 pin 10 (GND).

Follow these steps to verify functionality:

- Connect a CAT-5 Ethernet cable between an Ethernet switch and the RJ-45 connector on the CDK-8DUART
- Connect a PC to the same Ethernet switch
- Apply power to the CDK-8DUART
- Assign a static IP address to your network interface card on the PC
- Start the CobraNet Discovery application

You now should be able to monitor and manipulate the CDK-8DUART module using the CobraNet Discovery application. Connecting a second CobraNet device to the same network will allow audio and data transmission to be configured between the devices.

4.1 – Power Supplies

- Check power supplies to ensure CDK-8DUART is correctly powered
- 3.3V power needs to be within +/-10 % at the module pin

4.2 – Clocks and I2S Lines

- Check for the presence of clocks from the CDK-8DUART to the DACs and ADCs
- Check frequency of clocks. The default frequencies are LRCLK = 48kHz, SCLK = 3.072MHz and MCLK = 24.576MHz
- Check integrity of the clocks to ensure a reasonable square wave at the ADC and DAC (a jittery or very rounded clock signal will give poor audio performance)
- With the clocks correct, data should be seen on the incoming I2S lines to the CDK-8DUART (outgoing I2S lines will show nothing unless audio is routed to the local outputs)

4.3 – Ethernet LED Activity

Under normal operating conditions, the LED will work as follows.

- 1) With only a single CobraNet device on a network, both LEDs will be on and the left one will flash occasionally.
- 2) If a second CobraNet node is attached, or appropriate CobraNet software is running (such as CobraNet Discovery or Attero Tech Control Center), the left LED will begin flashing at around 3 times a second. The right LED shows whether the device is a conductor or not. If the light is on, the device is the conductor. If it is off, it is not the conductor. Only one device may be the conductor in a single CobraNet network.

APPENDIX A – CDK-8DUART SHMI Message Protocol

A1 – Terminology

The following table describes abbreviations, acronyms and technical terms used.

Term	Meaning
<i>Ack</i>	Positive Acknowledgement
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
HMI	Host Management Interface
SHMI	Serial Host Management Interface
IP	Internet Protocol
bps	Bits Per Second
MAC	Media Access Controller
MI	Management Interface
<i>Nack</i>	Negative Acknowledgement
SCI	Serial Communications Interface
SSI	Synchronous Serial Interface

A2 – System Description

The CDK-8DUART provides a straightforward CobraNet solution integration path, allowing the user to request information and set parameters via a serial port. The CDK-8DUART enables the user to easily add CobraNet capability to a given system. This system will use the 32-bit CobraNet Management Interface (MI) format. This system will enable a user to use an application (such as HyperTerminal, for example) to send American Standard Code for Information Interchange (ASCII) commands to request CobraNet information and set CobraNet parameters without having to know the details of the variable addresses

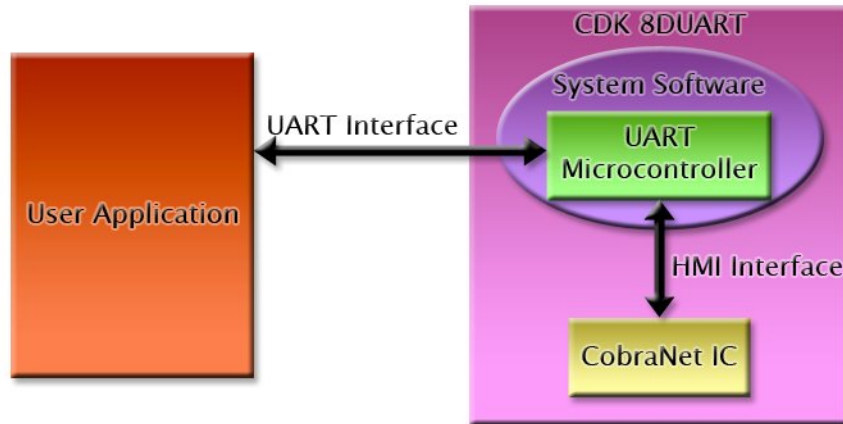


Figure A2-1 – System Block Diagram

The data flow starts from the user application. This can be any device capable of generating an RS232 serial signal such as a PC or a dedicated micro. The application generates a serial message containing a read or a write command. The CDK-8DUART translates the serial message and if valid, passes on the command to the CobraNet node via the HMI interface.

The CDK-8DUART then converts the response from the CobraNet node from the HMI into a serial message and passes the serial message back to the user application.

A3 – Serial Format

The CDK-8DUART interface configuration is set to 8 data bits, no parity bit, 1 stop bit, and no flow control at a default baud rate of 57,600 bits per second (bps). Of those parameters, only the baud rate can be changed. Acceptable values for the baud rate are 9600, 19200, 38400, 57600, and 115200 bps.

A3.1 – Message Format

The message protocol consists of a read message, a write message, and messages that can access a selected subset of the most commonly customized CobraNet MI variables. The messages all use ASCII characters to make it user-friendly and easy to use. Using ASCII also maximizes compatibility with applications that require printable ASCII characters like HyperTerminal, for example.

The messages can be characterized into the following main types:

- 1) Read variable messages.
- 2) Write variable messages.
- 3) Read response messages.
- 4) Write response messages.
- 5) Baud rate change message.

The read and write message formats follow the same basic structure. A read or write command is given first followed by the CobraNet MI variable data. Commands, and each subsequent data field, must be separated by a space character (hex value 0x20). The message is completed with a carriage return character (hex value 0x0D denoted by <CR>), a line feed character (hex value 0x0A denoted by <LF>), or both. Only a single value can be read or written at a time.

The variable being written or read can be referenced in two different ways. The first is to use the CobraNet MI variable name. The second method uses the variable’s address as an ASCII hexadecimal number (without the “0x” notation) such as 100400.

A list of the variable names supported can be found in Table A-12. The list also gives the variables hexadecimal addresses as well. Both methods are acceptable and both are shown in the following examples. However, it should be noted that only the most commonly used CobraNet MI variables are supported by name. If access to a non-supported variable is required, the address method must be used. All the address locations can be found in earlier sections of this document in the “Host Address” field of the CobraNet MI variable table.

There are a number of different data types when writing or reading data. The table below lists the data formats and the format in which they are read or written.

Data type	Format
String	“ABCDEF”
Integer	“33”
PhysAddress (MAC Address)	“01:02:03:04:05:06”
IPAddress	“1.2.3.4”

Table A-1 – Serial Message Data Types

A3.2 – Read Messages

The general read message structure conforms to the format specified in Table A-2 below.

Command		CobraNet MI Variable	Message Terminator
ReadVar	0x20	sysLocation	0x0D 0x0A
ReadAddr	0x20	100400	0x0D 0x0A

Table A-2 – Read Request Message Format

Example: ReadVar sysLocation<CR> or ReadAddr 100400<CR>

If the variable name is used, additional parameters may be required, depending on the variable being read. For instance, the variable may require a receiver number, as used by the rxBundle variable, or a receiver number and an audio channel number, as used by the rxSubMap variable. These parameters are simply added to the message after the variable name separated by a space.

Example: ReadVar rxSubMap 1 2<CR>

The order the parameters appear in is vitally important if a variable has more than one parameter. Information on which variables are accessible and which parameters are needed for each can be found in the variable's details section.

If the variable address is used, it is the only value required as each parameter of a CobraNet MI variable has its own address. Note that only sysDescr, sysContac, sysName, sysLocation, and ifPhysAddress of the MIB II MI variable addresses are implemented.

A3.2.1 – Read Message Response

If the variable is successfully read, the data is passed back to the user application. The format of the response is shown in Table A-3.

Message Command		CobraNet MI Variable		Value	Message Terminator
ReadVarRsp	0x20	sysLocation	0x20	“telephone closet, 3 rd floor”	0x0D 0x0A
ReadAddrRsp	0x20	100400	0x20	“telephone closet, 3 rd floor”	0x0D 0x0A

Table A-3 – Successful Read Response Message Formats

Example: ReadVarRsp sysLocation “telephone closet, 3rd floor”<CR> or
ReadAddrRsp 100400 “telephone closet, 3rd floor”<CR>

If the request fails for any reason, the response will be a *Nack* message. The format of the *Nack* message is shown in Table A-4.

Nack Reason Code			Message Terminator
0-9 & A-D	0x20	Nack	0x0D 0x0A

Table A-4 – Read *Nack* Message Format

Example: D Nack<CR>

The *Nack* response code gives an indication of the reason for the failure. A table of the reason codes and a description of each is shown in Table A-11.

A3.3 – Write Messages

The write message structure is very similar to the read message structure except it includes the new value contained within quotation marks. The format is specified in Table A-5.

Message Command		CobraNet MI Variable		New Value	Message Terminator
WriteVar	0x20	sysLocation	0x20	“telephone closet, 3 rd floor”	0x0D 0x0A
WriteAddr	0x20	100400	0x20	“telephone closet, 3 rd floor”	0x0D 0x0A

Table A-5 –Write Request Message Format

Example: WriteVar sysLocation “telephone closet, 3rd floor”<CR> or
WriteAddr 100400 “telephone closet, 3rd floor”<CR>

As with the read process, if the variable name is used then occasionally, depending on the variable being written, additional parameters may be required.

Example: WriteVar rxSubMap 1 2 “33”<CR>

Only the data to be written is contained within quotation marks. Again, as with the read process, if the address is used, it is the only value required as each parameter of a CobraNet MI variable has its own address.

A3.3.1 – Write Message Response

If the variable is successfully written, a simple *Ack* message is returned. The format of the response is shown in Table A-6.

	Message Terminator
Ack	0x0D 0x0A

Table A-6 – Successful Write Response Message Format

Example: Ack<CR>

If the write request fails, the response will be a *Nack* message. This message has the same format as the read *Nack* message and is repeated in Table A-7.

<i>Nack</i> Reason Code			Message Terminator
0-9 & A-D	0x20	Nack	0x0D 0x0A

Table A-7 – Write *Nack* Message Format

Example: D Nack<CR>

The *Nack* response code gives an indication of the reason for the failure. A table of the reason codes and a description of each is shown in Table A-11.

A3.4 – Baud Rate Change Message

Use this message when the baud rate of the device needs to be changed. When the baud rate change message is received by the CDK-8DUART, the change to the new baud rate will only occur if both of the following conditions are met:

1. The value is different from the current one
2. The value is a valid supported baud rate (9600, 19200, 38400, 57600, or 115200 bps).

Note that the baud rate value does not include a comma.

Message Command		New Baud Rate	Message Terminator
ChangeBaudRate	0x20	19200	0x0D 0x0A

Table A-8 – Baud Rate Change Request Message

Example: ChangeBaudRate 19200<CR>

If the baud rate change request satisfies both conditions noted above, the *Ack* response message (see Table A-9) will be transmitted at the original baud rate. Any further messages should then be sent at the new baud rate.

	Message Terminator
Ack	0x0D 0x0A

Table A-9 – Successful Baud Rate Change Message Format

Example: Ack<CR>

If either condition 1 or 2 above are not satisfied, the *Nack* response message will be sent. No baud rate change will occur and further messages will be processed at the original baud rate

Nack Reason Code			Message Terminator
0-9 & A-D	0x20	Nack	0x0D 0x0A

Table A-10 – Write *Nack* Message Format

Example: D Nack<CR>

A3.5 – *Nack* Response Descriptions

<i>Nack</i> Reason Code	<i>Nack</i> Reason
0	Receive buffer overflow (send commands slower)
1	Transmit buffer overflow (unable to send response quickly enough)
2	Unsupported CobraNet Address received
3	Invalid counter type, IP address, timer ticks value, MAC address, or attempt to write to read only value.
4	Attempt to read write only value
5	Unknown CobraNet Address received
6	Timeout waiting for CobraNet read response
7	Timeout waiting for CobraNet write response
8 – 9	Reserved
A – C	Reserved
D	Invalid Command Variable or Address
E – Z	Reserved

Table A-11 – *Nack* Reason Codes

A4 – Commands & Responses

The CDK-8DUART’s serial protocol processes a list of CobraNet MI variables by name. These are shown below.

Variable Name	Type	CobraNet Address (hexadecimal)
sysDescr	Read	100000
sysContact	Read/Write	100200
sysName	Read/Write	100300
sysLocation	Read/Write	100400
ifPhysAddress	Read	11000D
flashPersistEnable	Read/Write	1100
errorCode	Read	2002
errorCount	Read	2003
modeRateControl	Read/Write	2100
conductorStatus	Read	11000
serialFormat	Read/Write	24000
serialBaud	Read/Write	24001
serialRxMAC	Read/Write	24003
serialTxMAC	Read/Write	24100
rxSubFormat	Read/Write	4n30m (see section A4.16)
rxBundle	Read/Write	4n100 (see section A4.17)
rxSubMap	Read/Write	4n20m (see section A4.18)
txBundle	Read/Write	5n100 (see section A4.19)
txSubCount	Read/Write	5n105 (see section A4.20)
txUnicastMode	Read/Write	5n107 (see section A4.21)
txMaxUnicast	Read/Write	5n108 (see section A4.22)
txSubMap	Read/Write	5n20m (see section A4.23)
txSubFormat	Read/Write	5n30m (see section A4.24)
ipMonCurrentIP	Read/Write	72000
procMode	Read/Write	75100
stdUserVersionMajor	Read/Write	7E000
stdUserVersionMinor	Read/Write	7E001
stdUserString	Read/Write	7E100, 7E116, 7E12C, 7E142 (see section A4.29)
stdUserInteger	Read/Write	7E300 – 7E340 (see section A4.30)

Table A-12 – CobraNet MI Variable Names and their Addresses

The variable definitions for the above supported named variables can be found in sections A4.2 to A-19. Further details of all CobraNet MI variables are listed in the CobraNet Programmer’s Reference (by Cirrus Logic), Version 2.5. Only the 32-bit format definitions are to be used (as opposed to the available 24-bit format).

A4.1 – Legend

Variable Name	Name of variable
Description	Description of the variable including allowed values and usage discussion.
Host Address	HMI addresses are used to access variables via the host port.
Parameter	Only visible for variables that require one or more parameters when reading or writing using the variable name. It lists the parameters that are needed and the order they need to be in.
Size	<i>Size</i> is indicated for String <i>variables only</i> . It shows the number of characters available for that string.
Type	The data type of the variable. The following type Integer: a numeric value String: An ASCII string PhysAddress: A specifically formatted string to show the MAC address. Takes the form of “aa:bb:cc:dd:ee:ff” where aa, bb, cc, dd, ee, and ff are 2-digit hexadecimal values in the range 00 to FF. IP Address: A specifically formatted string to show the IP address. Takes the form of “a.b.c.d” where a,b,c, and d are decimal values between 1 and 254.
Attributes	Read-only variables can only be read and can not be modified. Read/Write variables can be read and written. Read/Write – Persistent variables can be read and written. If the persistence feature is enabled, values of these variables will automatically be written to flash for recall at startup.
Default	Value assigned to the variable at startup when persistence is disabled. The values of some Read-only variables reflect system conditions and thus may not have a default value. Note: The firmware in use may include an alternate default value.

A4.2 – sysDescr

Variable Name	sysDescr
Description	Describes type of interface as ASCII text. Format: <product specific description> CobraNet version <protocol version>.<major version>.<minor version>[.<manufacturer version>] <hardware platform><hardware rev> where <hardware platform> is the value of the firmwareHardwarePlatform device variable and <hardware rev> is the single digit revision number that is part of the Cirrus part number. Example: Attero Tech CDK-8 CobraNet version 2.11.6 CS496112
Host Address	100000
Size	84 characters
Type	String
Attributes	Read-only

A4.3 – sysContact

Variable Name	sysContact
Description	Contact details for the person responsible for this node.
Host Address	100200
Size	60 characters
Type	String
Attributes	Read/Write – Persistent
Default	Zero length string

A4.4 – sysName

Variable Name	sysName
Description	A name assigned to this managed node. By convention, this is the node's fully qualified domain name.
Host Address	100300
Size	60 characters
Type	String
Attributes	Read/Write – Persistent
Default Value	Product specific

A4.5 – sysLocation

Variable Name	sysLocation
Description	The physical location of this node (e.g., "telephone closet, 3 rd floor")
Host Address	100400
Size	60 characters
Type	String
Attributes	Read/Write – Persistent
Default Value	Zero length string

A4.6 – ifPhysAddress

Variable Name	ifPhysAddress
Description	The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack.
Host Address	11000D
Type	PhysAddress
Attributes	Read only
Default Value	N.A.

A4.7 – flashPersistEnable

Variable Name	flashPersistEnable
Description	Non-zero value enables variable persistence feature. Read/Write - Persistent type variables will be automatically written to non-volatile memory when changed. Values will be restored on power-up.
Host Address	1100
Type	Integer
Attributes	Read/Write - Persistent
Default Value	0

A4.8 – errorCode

Variable Name	errorCode
Description	Last error code reported.
Host Address	2002
Type	Integer
Attributes	Read only
Default Value	0

A4.9 – errorCount

Variable Name	errorCount
Description	Number of errors reported during system up time
Host Address	2003
Type	Integer
Attributes	Read only
Default Value	0

A4.10 – modeRateControl

Variable Name	modeRateControl
Description	Selects latency and sample rate mode for the interface. 1024 = 1 1/3 ms latency, 48 kHz sample rate 1281 = 1 1/3 ms latency, 96 kHz sample rate 1280 = 2 2/3 ms latency, 48 kHz sample rate 1537 = 2 2/3 ms latency, 96 kHz sample rate 1536 = 5 1/3 ms latency, 48 kHz sample rate 1793 = 5 1/3 ms latency, 96 kHz sample rate
Host Address	2100
Type	Integer
Attributes	Read/Write - Persistent
Default	1536

A4.11 – conductorStatus

Variable Name	conductorStatus
Description	Conductor status: 0 - This interface is not the conductor 1 - This interface is the conductor
Host Address	11000
Type	Integer
Attributes	Read Only

A4.12 – serialFormat

Variable Name	serialFormat
Description	<p>This variable is used to enable or disable the Serial Communications Interface (aka SCI or Serial Bridge) and to set the data format for both transmit and receive directions. Format may only be changed while the SCI is disabled. It is recommended to set <i>serialFormat</i> to 0, wait at least 100ms then set <i>serialFormat</i> to the desired value with the enable bit set. The SCI can take up to 100ms to recognize a change. This procedure ensures that the change is recognized and the port is properly configured.</p> <p>0x01 – Enable serial bridging. TXD pin is tri-stated when disabled. 0x02 – This bit is ignored. 0x04 – Use SCI_SCLK to control transmit enable for multi-drop (RS485) operation. SCI_SCLK is an active high signal; transmitter should be enabled when SCI_SCLK is high. 0x08 – Enable local loopback. This feature is intended primarily for factory test. SCI bridging must also be enabled for loopback to operate. When loopback is enabled, received characters are directed to the SCI transmitter instead of to the network <i>serialRxMac</i> should be set to 00:00:00:00:00:00 to avoid transmitter contention when loopback is enabled. 0x10 – Accept properly unicast addressed data in addition to data addressed in accordance to <i>serialRxMac</i> setting.</p>
Host Address	24000
Type	Integer
Attributes	Read/Write – Persistent
Default Value	0

A4.13 – serialBaud

Variable Name	serialBaud
Description	Baud rate for transmission and reception. The baud rate is specified in bits per second. The minimum baud rate is 600 baud. The maximum is 115200bps.
Host Address	24001
Type	Integer
Attributes	Read/Write – Persistent
Default Value	19200

A4.14 – serialRxMac

Variable Name	serialRxMAC
Description	MAC address of the CobraNet Interface from which SCI data will be accepted. Can only be changed when the SCI is disabled (serialFormat bit0 = 0). This may be any multicast address though 01:60:2B:FD:00:00 through 01:60:2B:FD:FF:FF have been reserved by Cirrus Logic for use as “asynchronous global channels.” <i>ifPhysAddress</i> is the only usable unicast address (CobraNet does not support Ethernet promiscuous mode).
Host Address	24003
Type	PhysAddress
Attributes	Read/Write – Persistent
Default Value	01:F0:2B:FD:00:00

A4.15 – serialTxMac

Variable Name	serialTxMAC
Description	MAC address of the CobraNet interface to which serial data is sent. May be any multicast or unicast address. serialTxMac only be changed when the SCI is disabled (serialFormat bit0 = 0)
Host Address	24100
Type	PhysAddress
Attributes	Read/Write – Persistent
Default	01:F0:2B:FD:00:00

A4.16 – rxSubFormat

Variable Name	rxSubFormat			
Description	Vector of received audio format for each sub-channel.			
	The least significant bit of these variables is set when the received format is supported for reception by the CobraNet interface. A test of this least significant bit can be used to determine correct reception on a per audio channel basis.			
	All entries in this vector will be 0 if rxStatus is zero.			
	Audio Format Value (decimal)	Resolution	Sample Rate	Latency
	0	No Signal		
	278,528	16 bit	48 kHz	5 1/3 ms
	344,064	20 bit	48 kHz	5 1/3 ms
	409,600	24 bit	48 kHz	5 1/3 ms
	1,343,488	16 bit	96 kHz	5 1/3 ms
	1,409,024	20 bit	96 kHz	5 1/3 ms
1,474,560	24 bit	96 kHz	5 1/3 ms	
270,336	16 bit	48 kHz	2 2/3 ms	

	335,872	20 bit	48 kHz	2 2/3 ms
	401,408	24 bit	48 kHz	2 2/3 ms
	1,327,104	16 bit	96 kHz	2 2/3 ms
	1,392,640	20 bit	96 kHz	2 2/3 ms
	1,458,176	24 bit	96 kHz	2 2/3 ms
	266,240	16 bit	48 kHz	1 1/3 ms
	331,776	20 bit	48 kHz	1 1/3 ms
	397,312	24 bit	48 kHz	1 1/3 ms
	1,318,912	16 bit	48 kHz	1 1/3 ms
	1,384,448	20 bit	48 kHz	1 1/3 ms
	1,449,984	24 bit	48 kHz	1 1/3 ms
Host Address	4n30m (n is 0 based receiver number, m is 0 based audio channel number)			
Parameters	1 – Receiver number (0-7) 2 – Audio sub-channel number(0-7)			
Type	Integer			
Attributes	Read only			

A4.17 – rxBundle

Variable Name	rxBundle
Description	Receive bundle assignment.
Host Address	4n100 (n is 0 based receiver number)
Parameters	1 – Receiver number (0-7)
Type	Integer16
Attributes	Read/Write – Persistent
Default Value	0

A4.18 – rxSubMap

Variable Name	rxSubMap
Description	Audio routing channel destinations for each audio channel in a received bundle.
Host Address	4n20m (n is 0 based receiver number, m is 0 based audio channel number)
Parameters	1 – Receiver number (0-7) 2 – Audio sub-channel number (0-7)
Count	8
Type	Integer
Attributes	Read/Write – Persistent

A4.19 – txBundle

Variable Name	txBundle
Description	Transmitter bundle assignment.
Host Address	5n100 (n is 0 based transmitter number)
Parameters	1 – Transmitter number (0-3)
Type	Integer16
Attributes	Read/Write – Persistent
Default Value	0

A4.20 – txSubCount

Variable Name	txSubCount
Description	Number of audio channels to transmit in a bundle.
Host Address	0x5n105 (n is 0 based transmitter number)
Parameters	1 – Transmitter number (0-3)
Type	Integer
Attributes	Read/Write – Persistent
Default Value	8

A4.21 – txUnicastMode

Variable Name	txUnicastMode
Description	<p>Specifies the number of unicast destinations served before automatically switching to multicast bundle transmission.</p> <p>0 – Multicast addressing used at all times. Note: multicast bundles do not transmit data until a receiver is assigned to the same bundle number.</p> <p>1 – 4 – Multicast addressing used to specify number of receivers.</p> <p>0x7FFFFFFF – Multicast addressing is never used. Maximum number of unicast destinations is set by <i>txMaxUnicast</i>. Receiver request priority is used to determine which receivers are serviced if multiple receivers are assigned to this bundle.</p>
Host Address	5n107 (n is 0 based transmitter number)
Parameters	1 – Transmitter number (0-3)
Type	Integer
Attributes	Read/Write – Persistent
Default Value	0x7FFFFFFF

A4.22 – txMaxUnicast

Variable Name	txMaxUnicast
Description	<p>Specifies maximum number of unicast destinations supported simultaneously by the transmitter. Receivers in excess of this setting will not receive the bundle.</p> <p>A transmitter can service up to four receivers. The number of unicast destinations transmitted to will never exceed this internal capacity limitation.</p> <p>If txUnicastMode is set lower than txMaxUnicast, the bundle will switch to multicast before the limitation on unicast destinations is reached.</p> <p>If txUnicastMode is set equal to txMaxUnicast, the bundle will switch to multicast when the limitation on unicast destinations is exceeded.</p>
Host Address	5n108 (n is 0 based transmitter number)
Parameters	1 – Transmitter number (0-3)
Type	Integer
Attributes	Read/Write – Persistent
Default Value	1

A4.23 – txSubMap

Variable Name	txSubMap
Description	Transmit audio channel (channel within bundle) to audio routing channel (channel of SSI) mapping. This vector contains the routing channel source specifiers per audio channel in the transmitted bundle.
Host Address	5n20m (n is 0 based transmitter number, m is 0 based sub-channel number)
Parameters	1 – Transmitter number (0-3) 2 – Audio sub-channel number (0-7)
Type	Integer
Attributes	Read/Write – Persistent
Default Value	First transmitter {1, 2, 3, 4, 5, 6, 7, 8} Second transmitter {1, 2, 3, 4, 5, 6, 7, 8}

A4.24 – txSubFormat

Variable Name	txSubFormat			
Description	Specifies data format for each sub-channel in the transmitted bundle. modeRateControl must also be set correctly to support the configured format.			
	Audio Format Value (decimal)	Resolution	Sample Rate	Latency
	0	No Signal		
	278,528	16 bit	48 kHz	5 1/3 ms
	344,064	20 bit	48 kHz	5 1/3 ms
409,600	24 bit	48 kHz	5 1/3 ms	

	1,343,488	16 bit	96 kHz	5 1/3 ms
	1,409,024	20 bit	96 kHz	5 1/3 ms
	1,474,560	24 bit	96 kHz	5 1/3 ms
	270,336	16 bit	48 kHz	2 2/3 ms
	335,872	20 bit	48 kHz	2 2/3 ms
	401,408	24 bit	48 kHz	2 2/3 ms
	1,327,104	16 bit	96 kHz	2 2/3 ms
	1,392,640	20 bit	96 kHz	2 2/3 ms
	1,458,176	24 bit	96 kHz	2 2/3 ms
	266,240	16 bit	48 kHz	1 1/3 ms
	331,776	20 bit	48 kHz	1 1/3 ms
	397,312	24 bit	48 kHz	1 1/3 ms
	1,318,912	16 bit	48 kHz	1 1/3 ms
	1,384,448	20 bit	48 kHz	1 1/3 ms
	1,449,984	24 bit	48 kHz	1 1/3 ms
Host Address	0x5n30m - 0x5n30m (n is 0 based transmitter number, m is 0 based sub-channel number)			
Parameters	1 - Transmitter number (0-3) 2 - Audio sub-channel number (0-7)			
Type	Integer			
Attributes	Read/Write - Persistent			
Default	0x54000			

A4.25 - ipMonCurrentIP

Variable Name	ipMonCurrentIP
Description	<p>The current IP address for the CobraNet interface. Changing the current IP address has an immediate affect on IP communications. A value of 0.0.0.0 indicates no IP address assignment for the interface. An IP address can be assigned (or reassigned) to the interface by any of the following means:</p> <p>A value loaded from <i>ipMonStaticIP</i> during power-up.</p> <p>A host processor writing to <i>pMonCurrentIP</i> via the SHMI.</p> <p>Receipt of a BOOTP response packet (typically in response to a transmitted BOOTP request)</p> <p>Receipt of a RARP response packet (RARP requests are not transmitted)</p>
Host Address	72000
Type	IP Address
Default Value	ipMonStaticIP
Attributes	Read/Write

A4.26 – procMode

Variable Name	procMode
Description	Signal processing mode: 0 - Silent 1 - Audio pass-through 2 - Running User DSP Configuration
Host Address	75100
Type	Integer
Attributes	Read/Write - Persistent

A4.27 – stdUserVersionMajor

Variable Name	stdUserVersionMajor
Description	Major user version number
Host Address	7E000
Type	Integer
Attributes	Read/Write - Persistent
Default Value	0

A4.28 – stdUserVersionMinor

Variable Name	stdUserVersionMinor
Description	Minor user version number
Host Address	7E001
Type	Integer
Attributes	Read/Write - Persistent
Default Value	0

A4.29 – stdUserString

Variable Name	stdUserString
Description	4 string variables to store the users string parameters.
Host Address	7E100 = User String 0 7E116 = User String 1 7E12C = User String 2 7E142 = User String 3
Parameter	1 - String number (0-3)
Size	60 characters
Type	String
Attributes	Read/Write - Persistent
Default Value	Zero length string

A4.30 – stdUserInteger

Variable Name	stdUserInteger
Description	64 integer variables to store the user's integer parameters.
Host Address	7E300 through 7E340
Parameter	1 - Integer Number (0-64)
Type	Integer
Attributes	Read/Write - Persistent
Default Value	0

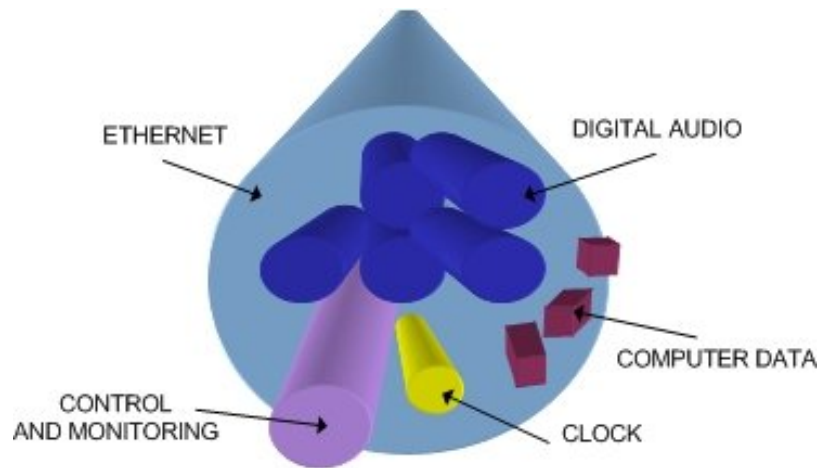
APPENDIX B – Introduction to CobraNet

CobraNet is an audio networking technology for delivery and distribution of real-time, high quality, uncompressed digital audio using a standard Ethernet network. It is implemented using a combination of hardware, firmware, and the CobraNet protocol.

Unlike other audio networking or distribution technologies, CobraNet is a true network and exists on standard Ethernet networks using standard Ethernet hardware. Since it is a true network, audio routing is highly flexible between network nodes and can be used in a variety of audio distribution applications.

In addition to the high degree of routing flexibility that CobraNet provides, the technology also incorporates the ability to monitor and control CobraNet devices remotely. This is a key feature that is highly important in fixed installation applications where the audio distribution equipment may not be readily accessible. All CobraNet devices on the network can be controlled and monitored from a central location by sending control commands and monitoring device specific parameters.

CobraNet provides this capability by implementing Simple Network Management Protocol (SNMP). SNMP is a standard protocol typically used for monitoring network devices such as Ethernet switches. In the case of CobraNet, it allows users to communicate with any CobraNet device using standard SNMP tools or a customized user interface designed specifically for CobraNet, such as Attero Tech’s Control Center application.



The figure above represents the types of data that coexist on a CobraNet network.

Before a CobraNet system can be configured, it is important to first understand how CobraNet distributes audio between devices.

Audio is sent in "bundles" on a CobraNet system. Each bundle is capable of holding up to 8 logical audio channels. Every CobraNet device has a number of bundle transmitters and bundle receivers. These transmitters and receivers are the mechanism used to send and receive bundles between devices.

For a transmitted bundle, audio may be sourced either directly from the local audio inputs of the device or from internal audio via the on-board DSP¹, but not both simultaneously. Internal audio from the onboard DSP could have originally been sourced from the local device inputs, sent from another CobraNet device or even generated by the DSP itself. Combinations of the local or internal audio may exist within a bundle in any order. Additionally, a single audio source in a device may be used multiple times in a single transmitter bundle or across multiple transmitter bundles.

For a received bundle, the received network audio may be routed directly to the device’s local outputs, the internal DSP¹ or simply ignored.

Once the contents of a bundle have been decided, the next step is to pass the bundle to another CobraNet device. To do this, every CobraNet device has up to 4 bundle transmitters. Each bundle transmitter has a transmit mode that must first be selected. This affects how many devices may receive that particular bundle at a time.

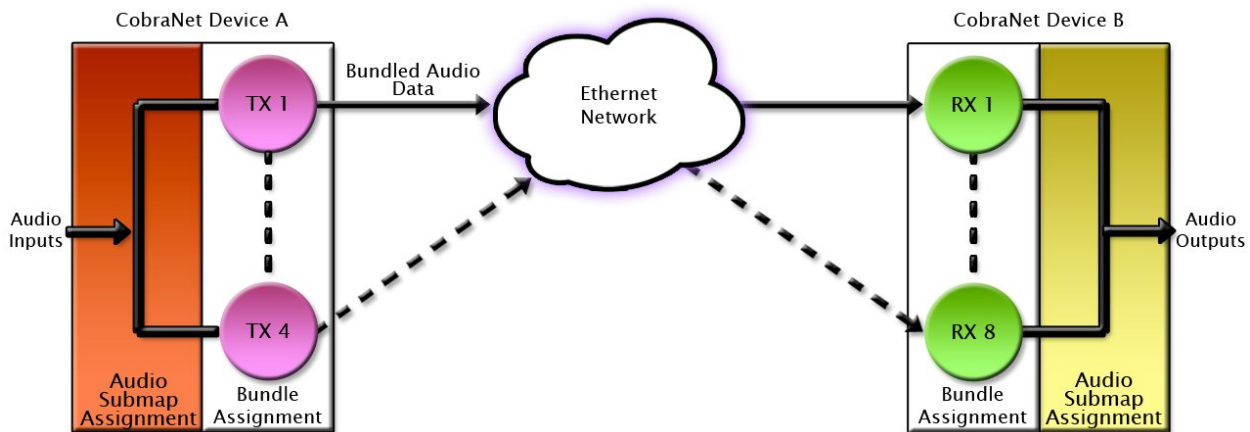
¹ Not available on all devices – CS496xxx devices only

The modes are as follows:

- Unicast – Used for one-to-one connections. In this mode, only one receiver at a time can receive this bundle. Once a link is established from this transmitted bundle to a receiver, any future requests for that bundle from other potential receivers will fail.
- Multicast – Used for one-to-many connections. This mode broadcasts its contents over the entire network. There is no restriction on the number of receivers. However, the downside is that CobraNet packets are distributed to all nodes on the network, whether they need them or not thus creating possible network bandwidth issues.
- Multi-unicasts – Another one-to-many mode. Whilst this is the most efficient method for getting a bundle to multiple receivers in terms of network bandwidth, it requires more processing power on the CobraNet device so in this mode there is a maximum limit of four receiver connections (this can be reduced if required). If more connections are required than the limit, the node can be configured to automatically switches to multicast.

Note: When a bundle must be transmitted to multiple receivers, multi-unicast transmissions should be used where possible.

Once the mode is selected, to enable a device to transmit the bundle, simply allocate the particular transmitter bundle a non-zero number. Since this number identifies all the network packets sent out by that transmitter, each transmit bundle number must be unique on a network².



Now that the transmitter is set up, it is time to set up the receivers. In order to receive bundles, each CobraNet device has up to eight bundle receivers. To enable a device to receive a bundle, simply allocate one of that device's bundle receivers the same bundle number as a transmitted bundle. By doing so, a virtual link is created and audio should now be passed from one device to the other. It should be noted that no knowledge of a device's network topology or connection is thus required in order to configure audio connections. The only restriction to this is that a device cannot be set up to receive a bundle it is also transmitting.

The above case creates a simple, one-to-one, unidirectional link. If more devices are required to receive that bundle, allocate the same transmitted bundle number to a bundle receiver on the other CobraNet devices.

It is also important to note that CobraNet supports simultaneous bidirectional audio distribution in each device. Not only could audio be sent from Device A to Device B but at the same time, should it be needed, audio could also be sent from Device B to Device A. The exact bundle and routing configuration will be determined by the needs of each individual installation. An installation may have multiple units transmitting multiple bundles. The only restriction is the bandwidth available on the network to transfer the audio.

CobraNet does more than just transfer audio data. It can be used to pass serial information as well. A feature called serial bridging has been incorporated that allows the passage of serial data between nodes. Each node can pass serial data to a specific node or multicast the data to multiple nodes. A node can also receive data from either a single source or multiple sources. Baud rates, data bits, stop bits, parity, and so on are all configurable. There is also support for multi-drop serial buses as well.

Finally, CobraNet has the capability to alter all of the above options in real time making the whole system completely dynamic. By use of control software, all of the bundle assignment parameters can be configured with no need to change cables, switch out connectors, or pull new wiring. Most importantly, this control capability can be implemented from a single location!

² Bundle numbers range from 1 through 65535. A value of 0 represents an inactive bundle. Numbers 1-255 are reserved for multicast mode transmissions only.

APPENDIX C – Reference Documents

The following table lists the relevant reference documents.

Document Title
CobraNet Programmer's Reference (Cirrus Logic)